ASPECTOS METODOLÓGICOS PARA RESOLVER PROBLEMAS DE SCHEDULING BASADOS EN UNA APLICACCIÓN REAL COMPLEJA

Area 2 – ATIA - Aspectos Teóricos de Inteligencia Artificial

Francisco S. Ibáñez, Daniel Díaz Araya y Raymundo Q. Forradellas
Laboratorio Integrado de Sistemas Inteligentes - Instituto de Informática –LISI/IdeI
Universidad Nacional de San Juan,
{ddiaz, fibanez, kike}@iinfo.unsj.edu.ar

Resumen

En este trabajo, se presentan aspectos metodológicos para resolver problemas de scheduling, obtenidos a partir de la experiencia en la resolución de problemas reales.
Inicialmente, se describe un problema real particular, que incluye varias características que se encuentran en otros problemas reales. Luego se presenta una breve reseña de un enfoque basado en restricciones, que se usará para modelar el problema. El enfoque usado combina las ventajas de la programación basada en restricciones con las de la programación orientada a objetos, siendo posible definir entre otras cosas, actividades, recursos y restricciones, entre actividades por un lado, y por otro lado entre actividades y recursos.
Finalmente, se muestra cómo se puede modelar el problema planteado mediante el enfoque anterior y se enfatizan aspectos metodológicos subyacentes a las decisiones del diseño del modelo.

1 Descripción del problema

La entrada al problema está constituida esencialmente por un conjunto de *pedidos*, efectuados por clientes de una empresa que produce el material necesario para fabricar envases de diferentes tipos de productos. A partir de estos pedidos se determina un conjunto de *actividades* que necesitan llevarse a cabo, mediante la utilización de recursos (considerando recursos tanto máquinas como insumos), a los efectos de obtener los productos finales requeridos por los clientes. Finalmente se establecen las restricciones que deben verificarse.
El objetivo de la aplicación es desarrollar una planificación de actividades y de recursos, de modo tal que se verifiquen todas las restricciones. Estas actividades deben realizarse en un orden determinado. Según el producto final que se desea obtener, estas actividades pueden usar diferentes máquinas. Las actividades, además de requerir máquinas (o recursos humanos) para poder llevarse a cabo, requieren insumos, por ejemplo, bobina con la materia prima, o productos semielaborados, por ejemplo, bobinas impresas. Una vez finalizadas las actividades, estas producen productos semielaborados que son utilizados por las actividades siguientes, o constituyen el producto final.
Para obtener un producto terminado, es necesario ejecutar la secuencia de actividades básicas, teniendo en cuenta las máquinas particulares que se utilicen, y considerando además los recursos que se requieren, antes de comenzar cada actividad, y los recursos producidos, luego de la finalización de las mismas.
Cada cliente realiza un pedido, determinando la cantidad de Kg. que se desea y especificando los detalles del producto final que permite determinar las máquinas que usarán las actividades básicas para obtener el mismo. Para cada pedido, se deben determinar las actividades necesarias para satisfacerlo. A tales efectos, las actividades se agrupan en tareas. Una Tarea, representa una secuencia de actividades básicas, que comienza con una actividad de impresión, a partir de una bobina virgen, y finaliza con una actividad de empaque, generando el producto final.
En función de la capacidad de la bobina virgen (medida en metros), se calcula la cantidad máxima de Kg., que produce una tarea. La cantidad de tareas necesarias para satisfacer el pedido, se determina en función de la cantidad de Kg. requerida, y de la cantidad máxima de Kg. que produce una tarea para el pedido correspondiente. Por ejemplo, si una empresa particular desea obtener 2500 Kg. de un determinado producto final, y cada tarea puede producir 1000 Kg. como máximo, serán necesarios tres tareas para completar el pedido.
La salida del problema, está constituida por una asignación de tiempos de comienzo y de finalización de cada una de las actividades involucradas en la representación del problema.
El problema ha sido resuelto optimizando el tiempo de finalización, para algunos casos, y conformándose con tiempos subóptimos para casos más complejos. Sin embargo, en el presente trabajo, sólo consideraremos los aspectos metodológicos inherentes a la representación del problema.

2 Breve reseña de la Metodología utilizada

En programación con restricciones, un problema se representa esencialmente, en términos de variables que involucran incertidumbre, y restricciones que deben verificar estas variables.
Una vez representado el problema, la resolución del mismo consiste en encontrar valores de cada una de las variables, de modo que se verifiquen todas las restricciones [Ddiaz,98]. Finalmente, se debe considerar el criterio de optimización para elegir una solución.
En este trabajo usaremos un enfoque basado en [ISch-R,99], [ISch-U,99], que combina esencialmente programación con restricciones con programación orientada a objetos, aunque los conceptos subyacentes a este trabajo no se restringen a ser implementados con este enfoque. En este enfoque, se parte de clases, definidas para representar problemas de scheduling. Las clases más usadas son aquellas para representar actividades y recursos. Para representar un problema particular, se deben crear instancias de actividades y de recursos y se deben incluir las restricciones que impone la aplicación particular, mediante la utilización de funciones miembros que proveen estas clases. Una actividad puede requerir un recurso, en cuyo caso la instancia de la clase que representa el recurso debe especificar que el recurso será *requerido*, o puede proveer un recurso, en cuyo caso, la instancia debe especificar que el recurso será *provisto*.
Una de las principales ventajas de este enfoque, es que mediante su utilización, es posible representar explícitamente un problema de scheduling. Tal representación sirve además como una especificación declarativa. Finalmente, a la hora de resolver un problema, es posible confiar en las primitivas de control del resolvedor de restricciones (CS – *Constraints Solver*), sin tener necesidad de conocer cómo trabaja internamente el mecanismo de propagación de restricciones.

3 Representación del problema propuesto.

Como se comentó en el apartado anterior, la resolución del un problema de scheduling se divide básicamente en tres partes, la representación del problema, la búsqueda de soluciones, y el criterio de optimización. En el presente trabajo, consideraremos aspectos metodológicos inherentes a la representación del problema. En esta aproximación usamos programación orientada a objetos. Sin embargo, antes de describir las clases y las instancias de las mismas, es necesario reflexionar en algunas consideraciones relevantes en cuanto al diseño del modelo.
En primer lugar es importante distinguir la necesidad o no de usar recursos. Las máquinas que usan las actividades se modelan como recursos requeridos. Dependiendo de cuántas máquinas con las mismas características existan, los recursos utilizados para modelarlas serán unarios o discretos. Por ejemplo, hay actividades que puede requerir tres tipos diferentes de máquinas:
En principio, no sería estrictamente necesario usar recursos para el caso de que varias actividades utilicen una sola máquina. Sin embargo, en la práctica es necesario debido a razones de eficiencia.
Considere el caso de tres actividades, que usan una misma máquina y que se deben planificar a partir del tiempo *t0*, y debe finalizar en un tiempo menor o igual a *t1*. Se podría modelar definiendo tres instancias *a1, a2* y *a3,* que representen las actividades e incluyendo restricciones disyuntivas entre *a1* y *a2*, entre *a1* y *a3*, y entre *a2* y *a3*. Las restricciones disyuntivas se pueden modelar mediante subobjetivos que contengan las dos alternativas de la disyunción, de modo que, al ejecutarse el programa, se incluye la primer alternativa en el almacenamiento de restricciones, y si se produce un fallo, se realiza backtracking, se restaura la base de restricciones y se incluye la segunda alternativa.
Sin embargo, este modo de modelar las restricciones disyuntivas, puede ser muy ineficiente. Como ejemplo, considere que las actividades tienen una duración igual a 4, que *t0* es igual a *0*, y que *t1* es igual a 11. Para detectar inconsistencia, es necesario considerar las alternativas de las disyunciones, y verificar, mediante backtracking, que no existe ninguna solución. Modelando el problema mediante recursos unarios, el problema de representa en forma más natural, por un lado, y por otro lado, es posible, usando restricciones adecuadamente, detectar inconsistencia sin necesidad de backtracking, lo cual es muy importante desde el punto de vista de la eficiencia. La modelización de restricciones disyuntivas mediante la utilización de recursos unarios, no siempre evita el backtracking. Dependiendo del enfoque particular que se esté usando y de las restricciones que se incluyan, el backtracking puede evitarse o no. Por ejemplo, en [ISch-R,99], [ISch-U,99], existen dos funciones miembros para la clase que representa recursos unarios, una que genera restricciones disyuntivas entre cada par de actividades, y otra que "cierra" el recurso, imponiendo la restricción que expresa que en cada instante de tiempo, sólo una actividad puede usar el recurso. La utilización de la primera función miembro, no

detecta la inconsistencia para el caso planteado anteriormente, y es necesario agregar la segunda para lograrlo. La razón de lo anterior se basa en el hecho de que la primera función impone una restrición disyuntiva a cada par de actividades, pero no tiene en cuenta el conjunto de actividades como un todo. Consecuentemente, la inconsistencia no se detecta, puesto que para cada par de actividades, consideradas aisladamente, existe al menos una solución que verifica la restricción disyuntiva.
Finalmente, para el caso de la actividad de empaque, existe suficiente personal para llevar a cabo la misma, y por lo tanto no es necesario representar los recursos humanos. En otras palabras, no es necesario representar en el modelo, el hecho de que la actividad de empaque requiere recursos humanos. Solamente cuando los recursos son limitados, como en los casos anteriormente descriptos, el modelo debe reflejar los requerimientos de recursos.
En el trabajo se profundiza sobre el modelado de los recursos requeridos y provistos, el modelado de insumos, productos semielaborados y productos finales, la descripción de las estructuras de datos, clases e instancias y las consideraciones de tiempos adicionales entre actividades consecutivas

4 Conclusiones

En el presente trabajo, se ha descripto un problema de scheduling, y se ha aplicado un enfoque basado en restricciones y en programación orientada a objetos para representar el problema planteado.
La experiencia obtenida en el desarrollo de esta aplicación, podría sintetizarse esencialmente en tres puntos. Por un lado, la resolución de este tipo de problemas es prácticamente inviable sin disponer de un CS. El CS, no necesariamente debe ser el propuesto por [ISol-R,99], podría ser el desarrollado en el Lenguaje *CHIP* (Constraint Handling In Prolog) [Hente,89], usado dentro del marco de la programación lógica, y aplicado a problemas de combinatoria discreta [Rueda,95], el desarrollado en *mOZart* [Denys,98], usado en un marco que, además de incorporar características nuevas, conserva las características fundamentales de los paradigmas funcionales, de la programación lógica y de la programación orientada a objetos, o bien podría usarse cualquier otro enfoque basado en restricciones [Ibañez,94], [Forradel,99].
El CS permite, incluir de un modo declarativo las restricciones, confiando en que el mecanismo subyacente al CS realice la propagación de restricciones, eliminado las ramas del árbol de búsqueda que no conducen a solución alguna y reduciendo consecuentemente el árbol de búsqueda. Sin embargo, aunque en muchos casos no es necesario conocer en profundidad los mecanismos internos subyacentes al CS, es esencial entender el comportamiento del mismo, para reducir efectivamente el tamaño del árbol de búsqueda. Por ejemplo, tratando restricciones disyuntivas sin la utilización de recursos unarios, o con recursos unarios pero sin explotar toda su potencia, se reduce el tamaño del árbol de búsqueda generado por las variables dominio que representan los comienzos de las actividades, pero no se reduce tamaño el árbol generado por las alternativas de las disyunciones.
Finalmente, las ventajas proporcionadas por la programación orientada a objetos, combinadas con aquellas derivadas de la programación con restricciones, permiten desarrollar aplicaciones muy complejas en tiempos razonables.

5 Referencias

[Forradel,99] R. Forradellas, F. Ibañez y R. Berlanga, "Un modelo para el tratamiento de Sistemas Dinámicos basado en la Satisfacción de Restricciones", WICC99, UNSJ, 1999.
[Hente,89] Pascal Van Hentenryck, "Constraints Satisfaction in Prolog", MIT Press, 1989
[Ibañez,94] F. Ibañez, "CLP(Temp), Integración de Restricciones Temporales Métricas y Simbólicas, en el Marco CLP", Tesis Doctoral, Universidad Politécnica de Valencia, España, 1994.
[ISol-R,99] "Ilog Solver - Reference Manual", Ilog, France, 1999.
[ISol-U,99] "Ilog Solver - User Manual", Ilog, France, 1999.
[ISch-R,99] "Ilog Schedule- Reference Manual", Ilog, France, 1999.
[Rueda,95] Rueda L. Klenzi R. Gutierrez L. Ibañez F. Forradellas R., "Tratamiento de Problemas de Combinatoria Discreta mediante el Paradigma CLP", ATIA, UNS, Bahía Blanca, 1995.
[Ddiaz,98] Daniel Diaz Araya - Aplicación de Tecnologías Basadas en Restricciones a la Resolución de Problemas Industriales - Tesis de Grado , Universidad Nacional de San Juan, 1998.
[Denys ,98] Denys Duchier, Leif Kornstaedt, Christian Schulte, and Gert Smolka. "A Higher-order Module Discipline with Separate Compilation, Dynamic Linking, and Pickling". Technical Report, Programming Systems Lab, DFKI and Universität des Saarlandes, 1998.